REMARKS

With this amendment, claims 1-33 and 35-39 remain pending in the present application. Claim 14 has been amended to correct a typographical error. Claims 1, 7, 9, 14, 16, 22, 24, 26, 27, 30, and 37-39 have been amended to more particularly claim the invention. Claim 33 has been amended to include subject matter in original claim 34, and claims 35-37 have been amended accordingly. Claim 34 has been canceled. The Applicant has carefully and thoughtfully considered the Office Action and the comments therein. For the reasons given below, it is submitted that this application is in condition for allowance.

1. Rejections under 35 USC § 103 - Mantooth in view of Zink

On pages 2-5, the Action rejects claims 31 and 33-39 under 35 U.S.C. § 103(a) as being unpatentable over U.S. Patent No. 6,236,956 to Mantooth et al. (hereinafter Mantooth) in view of U.S. Patent No. 6,738,964 to Zink et al. (hereinafter Zink). Applicants respectfully traverse the rejection. Mantooth and Zink are first discussed, followed by a discussion of the instant application. Thereafter, the rejections of the claims are discussed.

A. U.S. Patent No. 6,236,956 to Mantooth et al.

Mantooth discloses a method of using a computer to simulate analog and mixed-signal circuits and systems. Mantooth, col. 1, lines 12-14. A known code generator is used to convert a model into AHDL code. Mantooth, claim 6, lines 32-36. The AHDL code is then input into a simulator for simulating the model system. Mantooth, col. 6, lines 36-38. Mantooth aims "to improve simulation robustness by providing tools that allow a non-expert user to improve simulation performance, identify and resolve simulation problems." Mantooth, col. 2, lines 59-62. Two such tools are the Parameter Manager and the Sample Point Assistant. Mantooth, col. 3, lines 64-67 and col. 4, lines 6-10.

In Mantooth, a parameter is described as "a facet of a model that has a value that generally is held constant during a particular simulation. Mantooth, col. 10, lines 2-5. While parameters are

automatically given default values for range of validity, initial value, and comment, the user is free change these values. Mantooth, col. 10 lines 51-55 and FIG. 10. The Parameter Manager, therefore, is an "interactive tool" wherein the user can manually "inspect[] and specify[] valid operating regions, default values, and descriptions of parameters in a model." Mantooth, col. 3, lines 64-67 and col. 10, lines 11-18.

The Sample Point Assistant is "[a]n interactive, graphical tool for identifying non-linear dependencies in a model, and for selecting, inspecting and modifying a series of sample points for a selected non-linear relationship." Mantooth, col. 4, lines 6-10. Specifically, the Sample Point Manager is a visualization tool that facilitates sample point selection, inspection and modification in an interactive, graphical mode." Mantooth, col. 12, lines 21-24. This allows a plot of a model parameter to be modified, automatically by the Sample Point Assistant or manually by the user through the Sample Point Assistant, as opposed to automatically adjusting the parameter itself. Mantooth, col. 11, lines 59-60, col. 12, lines 8-11, 17-21, 31-33, and col. 3, lines 64-67.

B. U.S. Patent No. 6,738,964 to Zink et al.

Zink discloses a system and method for graphical development of software and software/hardware hybrid systems. Zink, col. 1, lines 43-44. These systems may be represented by drawings which are comprised of user defined components. Zink, col. 13, lines 48-49 and 57-59. After the user creates the drawing, but before the drawing runs for the first time, the user must configure the components. Zink, col. 13, lines 60-62. The user may access the property settings via "'wizards' that prompt the user for performance-related information and then process the user inputs to determine actual property settings." Zink, col. 13, line 66, to col. 14, line 5.

In addition, Zink teaches including scripting language in components. Zink, col. 23, lines 53-55. "Scripting provides capabilities for achieving compatibility (i.e. interoperability) as well as optimization based on user-defined goals." Zink, col. 25, lines 1-3. "The scripting language permits the expert (who creates a component) to embed a significant measure of his (or her) integration knowledge into the component. Based on the script's ability to interrogate the operating

environment, the expert can embed integration knowledge to select (or build) the appropriate code modules and contribute them to the solutions project." Zink, col. 23, lines 60-63.

C. U.S. Patent Application No. 10/731,548

The instant invention relates to the generation of code from a graphical model, such as the exemplary block diagram in Figure 16A. See, e.g., specification, page 1, lines 4-5; page 39, lines 8-9. As an example, in step 130 of Figure 15, a user, in response to a query by a wizard, specifies general code generation goals. See, e.g., specification, page 41, lines 15-23. In Figure 17, an exemplary graphical user interface 1000 includes a general code generation goals pane 200, which provides exemplary fields 210-224 for the user to specify the general code generation goals of a graphical model. See, e.g., specification, page 43, line 9, to page 44, line 22. In block 140 of Figure 15, the wizard configures the parameters within the graphical model based on the specified general code generation goals. See, e.g., specification, page 41, lines 24-25. In Figures 18A-18B, an exemplary wizard presents to the user a list of the parameters of each object in the graphical model that were changed and not changed in response to the specified general code generation goals. See, e.g., specification, page 44, lines 24-28. In block 150 of Figure 15, the wizard prompts the user to select checks and analysis that should be performed on the graphical model prior to code generation to ensure that the graphical model is suitable for generating code and complying with the specified general code generation goals. See, e.g., specification, page 41, line 27, to page 42, line 6. In Figures 19A-19E, an exemplary model advisor pane 320 lists a number of conditions that can be checked based on the specified general code generation goals. See, e.g., specification, page 45, line 1, to page 48, line 20. In block 190 of Figure 15, code is generated for the model that is consistent with the specified general code generation goals. See, e.g., specification, page 42, line 31, to page 43, line 2.

D. Claims 31 and 33-39

Claim 31 recites, "[i]n a graphical modeling environment, a medium holding computer-executable instructions for a method, comprising the steps of: displaying a user interface for

prompting a user to specify one or more code generation goals; and automatically changing parameters of the graphical model that are inconsistent with the code generation goals specified by the user."

In rejecting claim 31, the Office Action erroneously aligns the recited "automatically changing parameters of the graphical model that are inconsistent with the code generation goals specified by the user" with the Parameter Manager and the "automatic or manual methods for modifying the sample points" of Mantooth. Mantooth, col. 12, lines 31-33 and col. 3, lines 64-67. As stated in the Office Action, Mantooth fails to teach "displaying a user interface for prompting a user to specify one or more code generation goals." Applicants agree. To overcome the failings of Mantooth, the Office Action relies on the teachings of Zink. In particular, the Office Action erroneously aligns the recited "user interface" with the "wizards" of Zink that prompt the user for performance related information and then process the user inputs to determine actual property settings and "optimization based on user-defined goals." Zink, col. 14, lines 2-5 and col. 25, lines 2-3.

The Office Action, however, fails to establish a <u>prima</u> <u>facie</u> case of obviousness because the combination of Mantooth in view of Zink does not teach the claimed invention for at least the following three reasons. First, Mantooth fails to teach or suggest "automatically changing parameters." Second, Mantooth fails to teach or suggest changing parameters "that are inconsistent with the code generation goals specified by the user." Third, Zink fails to teach or suggest "displaying a user interface for prompting a user to specify one or more code generation goals."

First, Mantooth fails to teach or suggest "<u>**automatically changing**</u> parameters of the graphical model that are inconsistent with the code generation goals specified by the user," as recited in claim 31. As discussed above, Mantooth attempts "to improve simulation robustness by providing tools that allow a non-expert user to improve simulation performance, identify and resolve simulation problems." Mantooth, col. 2, lines 59-62. Mantooth teaches a "Parameter Manager," wherein the parameters are automatically set to default values. Mantooth, col. 10 lines 51-53. The user is then free to adjusted these values <u>**manually**</u>, not <u>**automatically**</u>. Mantooth, col. 3, lines 64-67, col. 10, lines 53-55, FIG. 10. The "Parameter Manager," therefore, is an

"**interactive tool** for **inspecting** and **specifying** valid operating regions, default values, and descriptions of parameters in a model." Mantooth, col. 3, lines 64-67. Mantooth therefore teaches a user **manually** interacting with the parameter manager by inspecting and then specifying the operating regions, default values, and descriptions of parameters to be used in the model. Mantooth, col. 3, lines 64-67 and col. 10, lines 11-18. In contrast, claim 31 requires "**automatically changing** parameters of the graphical model that are inconsistent with the code generation goals specified by the user." The Office Action also cites the use of the word "automatic" in Mantooth by quoting "automatic or manual methods for modifying the sample points." Mantooth, col. 3, lines 64-67. However, "automatic or manual methods for modifying the sample points" is for displaying a plot of a parameter and automatically adjusting the sample points in the plot, as opposed to automatically adjusting the parameter itself. Mantooth, col. 11, lines 59-60, col. 12, lines 8-11, 17-21, 31-33, and col. 3, lines 64-67. Zink fails to overcome the failings of Mantooth. Hence, the combination of Mantooth and Zink fails to teach "**automatically changing** parameters of the graphical model that are inconsistent with the code generation goals specified by the user."

Second, Mantooth fails to teach or suggest "automatically changing parameters of the graphical model **that are inconsistent with the code generation goals specified by the user**," as recited in claim 31. As discussed above, Mantooth attempts "to improve simulation robustness by providing tools that allow a non-expert user to improve simulation performance, identify and resolve simulation problems." Mantooth, col. 2, lines 59-62. However, as previously addressed, the "Parameter Manager" only allows the user to "**inspect**[] and **specify**[] valid operating regions, default values, and descriptions of parameters in a model." Mantooth, col. 3, lines 64-67 and col. 10, lines 11-18. The "Parameter Manager" of Mantooth does not include the ability to automatically identify and then change parameters that are inconsistent with the code generation goals specified by the user. **The Parameter Manager of Mantooth, therefore, lacks any discussion of parameters that are inconsistent with code generation goals.** In contrast, claim 31 requires "automatically changing parameters of the graphical model **that are inconsistent with the code generation goals specified by the user**." Further, Zink fails to overcome the failings of Mantooth. Hence, the combination of Mantooth and Zink fails to teach "automatically changing

parameters of the graphical model **that are inconsistent with the code generation goals specified by the user**."

Third, Zink fails to teach or suggest "displaying a user interface for prompting a user to specify **one or more code generation goals**," as recited in claim 31. As discussed above, Zink discloses a system and method for graphical development of software and software/hardware hybrid systems. Zink, col. 1, lines 43-44. These systems may be represented by drawings which are comprised of components. Zink, col. 13, lines 48-49 and 57-59. However, in contrast to the claimed invention, Zink teaches **a user configuring the property settings for each component** on the drawing after the user creates the drawing, but before the drawing runs for the first time. Zink, col. 13, lines 60-62. "User access to property settings may be presented to the user in several different ways including [] 'wizards' that prompt the user for performance-related information and then process the user inputs to determine actual property settings." Zink, col. 13. line 66, to col. 14, line 5. In addition, Zink teaches including scripting language in components. Zink, col. 23, lines 53-55. "The scripting language permits the expert (who creates a component) to embed a significant measure of his (or her) integration knowledge into the component. Based on the script's ability to interrogate the operating environment, the expert can embed integration knowledge to select (or build) the appropriate code modules and contribute them to the solutions project." Zink, col. 23, lines 60-63. **Zink, therefore, teaches configuring the property settings for each individual component via a wizard or scripting language**, as opposed to "displaying a user interface for prompting a user to specify **one or more code generation goals**," as recited in claim 31. Mantooth fails to overcome the failings of Zink. Hence, the combination of Mantooth and Zink fails to teach "displaying a user interface for prompting a user to specify **one or more code generation goals**," as recited in claim 31.

Therefore, Mantooth and Zink fail to teach or fairly suggest claim 31.

Claim 33 recites, "[a]n apparatus comprising: at least one processor; a memory coupled to the at least one processor; and a computer program residing in the memory and being executed by the at least one processor, wherein the computer program includes a wizard for guiding a user

through a process for preparing a graphical model for a code generation process for creating code based on the graphical model and at least one code generation goal specified by the user."

In rejecting claim 33, the Office Action asserts that Mantooth fails to teach a wizard for guiding a user through a process. Applicants agree. To overcome the failings of Mantooth, the Office Action relies on the teachings Zink. In particular, the Office Action erroneously aligns the recited "wizard for guiding a user through a process for preparing a graphical model for a code generation process for creating code based on the graphical model and at least one code generation goal specified by the user" with the "icon-based graphical user interfaces, dedicated property dialog windows, and via 'wizards' that prompts the user for performance related information and then process the user inputs to determine actual property settings" of Zink. Zink, col. 14, lines 2-5.

The combination postulated by the Office Action, however, fails to establish a prima facie case of obviousness because the combination of Mantooth in view of Zink does not teach the claimed invention for at least the following reason.

Zink fails to teach or suggest "a wizard for guiding a user through a process for preparing a graphical model for **a code generation process for creating code based on** the graphical model and **at least one code generation goal specified by the user**," as recited in claim 33. As discussed above, Zink discloses a system and method for graphical development of software and software/hardware hybrid systems. Zink, col. 1, lines 43-44. These systems may be represented by drawings which are comprised of components. Zink, col. 13, lines 48-49 and 57-59. However, Zink teaches **configuring the property settings for each component** on the drawing after the user creates the drawing, but before the drawing runs for the first time. Zink, col. 13, lines 60-62. "User access to property settings may be presented to the user in several different ways including [] 'wizards' that prompt the user for performance-related information and then process the user inputs to determine actual property settings." Zink, col. 13. line 66, to col. 14, line 5. **Zink, therefore, teaches configuring the property settings for each component via a wizard**, as opposed to "a wizard for guiding a user through a process for preparing a graphical model for **a code generation process for creating code based on the graphical model and at least one code generation goal specified by the user**," as recited in claim 33. Mantooth fails to overcome the failings of Zink.

Hence, the combination of Mantooth and Zink fails to teach "a wizard for guiding a user through a process for preparing a graphical model for **a code generation process for creating code based on** the graphical model and **at least one code generation goal specified by the user**," as recited in claim 33.

Dependent claims 34-39 are allowable, at least, for being dependent from an allowable claim.

2. Rejections under 35 USC § 103 - Mantooth in view of Zink and Iborra

On pages 5-19, the Action rejects claims 1-30 and 32 under 35 U.S.C. § 103(a) as being unpatentable over Mantooth in view Zink and U.S. Patent No. 7,137,100 to Iborra et al. (hereinafter Iborra). Applicants respectfully traverse the rejection. Mantooth, Zink, and the instant application are discussed above. Iborra is first discussed, followed by a discussion of the claims.

A. U.S. Patent No. 7,137,100 to Iborra et al.

Iborra discloses an automated software production system. Iborra, abstract. This system includes front end processing to create a concept model. Iborra, col. 3, lines 54-55. The conceptual model is created "using a graphical user interface to represent the various objects, etc. that comprise the model visually. Iborra, col. 4, lines 48-51. The "concept model is typically written as statements in any known or new formal language which has rules of syntax and semantics (together referred to as grammar). Based on these rules, the concept model can be validated to make sure it is syntactically complete, correct and not ambiguous." Iborra, col. 3, lines 57-60 and 64-66. A translator program may then be used to write a compelte working program. Iborra, col. 5, lines 1-6.

B. Claims 1-30 and 32

Claim 1 recites, "a method for generating embedded code from a model, comprising the steps of: prompting a user to specify at least one code generation goal for the embedded code; and generating code in a compliable form for the specified code generation goal."

In rejecting claim 1, the Office Action asserts that both Mantooth and Zink fail to teach, at least, a method including generating code in a compliable form for the specified code generation goal. Applicants agree. To overcome the failings of Mantooth and Zink, the Office Action relies on the teachings Iborra. In particular, the Office Action erroneously aligns the recited "generating code in a compliable form for the specified code generation goal" with "other species may start at the first statement and process it to make sure it complies with every applicable rule and then repeat this process for every other statement" of Iborra. Iborra, col. 6, lines 15-18.

The three-way combination postulated by the Office Action, however, fails to establish a prima facie case of obviousness for at least the following two reasons. First, Zink fails to teach or suggest "prompting a user to specify at least one code generation goal." Second, Iborra fails to teach or suggest "generating code in a compliable form for the specified code generation goal."

First, Zink fails to teach or suggest "prompting a user to specify **at least one code generation goal,**" as recited in claim 1. As discussed above, Zink discloses a system and method for graphical development of software and software/hardware hybrid systems. Zink, col. 1, lines 43-44. These systems may be represented by drawings which are comprised of components. Zink, col. 13, lines 48-49 and 57-59. In contrast to the claimed invention, Zink teaches **the user configuring the property settings for each component** on the drawing after the user creates the drawing, but before the drawing runs for the first time. Zink, col. 13, lines 60-62. "User access to property settings may be presented to the user in several different ways including [] 'wizards' that prompt the user for performance-related information and then process the user inputs to determine actual property settings." Zink, col. 13. line 66, to col. 14, line 5. In addition, Zink teaches including scripting language in components. Zink, col. 23, lines 53-55. "Scripting provides capabilities for achieving compatibility (i.e. interoperability) as well as optimization based on user-defined goals. Specific capabilities (enabled via scripting) that may be useful include elimination of parameter range-checking, elimination of data format conversions, re-linking selected code modules, patching code fragments into object modules, etc." Zink, col. 25, lines 1-7. "The scripting language permits the expert (who creates a component) to embed a significant measure of his (or her) integration knowledge into the component. Based on the script's ability to interrogate the

operating environment, the expert can embed integration knowledge to select (or build) the appropriate code modules and contribute them to the solutions project." Zink, col. 23, lines 60-63. **Zink, therefore, teaches configuring the property settings for each individual component via a wizard or scripting language**, as opposed to "prompting a user to specify **at least one code generation goal**," as recited in claim 1. Mantooth and Iborra fail to overcome the failings of Zink. Hence, the combination of Mantooth, Zink, and Iborra fails to teach "prompting a user to specify **at least one code generation goal**," as recited in claim 1.

Second, Iborra fails to teach or suggest "generating code **in a compliable form for the specified code generation goal**," as recited in claim 1. As discussed above, the conceptual model is created "using a graphical user interface to represent the various objects, etc. that comprise the model visually. Iborra, col. 4, lines 48-51. A translator program may then be used to write a complete working program. Iborra, col. 5, lines 1-6. In contrast to the claimed invention, Iborra teaches a "concept model [which] is typically written as statements in any known or new formal language which has rules of syntax and semantics (together referred to as grammar). Based on these rules, the concept model can be validated to make sure it is syntactically complete, correct and not ambiguous." Iborra, col. 3, lines 57-60 and 64-66. **Iborra, therefore, teaches validating code based on a set of rules, as opposed to validating code based on specified code generation goals**. In contrast, claim 1 requires "generating code in a compliable form for the specified code generation goal." Mantooth and Zink fail to overcome the failings of Iborra. Hence, the combination of Mantooth, Zink, and Iborra fails to teach "generating code **in a compliable form for the specified code generation goal**," as recited in amended claim 1.

Dependent claims 2-13 are allowable, at least, for being dependent from an allowable claim.

Independent claim 14 recites similar subject matter to that recited in claim 31, which is allowable over Mantooth in view of Zink as discussed above. In addition, Iborra fails to overcome the failings of Mantooth and Zink. Therefore, claim 14 is allowable for the same reasons discussed above in connection with claim 31.

Dependent claim 15 is allowable, at least, for being dependent from an allowable claim.

Claim 16 recites, "[a] method of preparing a model for embedded code generation, the method comprising the steps of: displaying a graphical user interface though which a user can specify at least one code generation goal for the embedded code to be generated from the model; and in response to a user specifying a code generation goal, providing feedback to the user regarding compliance of the graphical model with said code generation goal."

In rejecting claim 16, the Office Action states that Mantooth fails to teach "that the code is embedded, that generated code is in a compliable form, prompting a user to specify at least one code generation goal for the embedded code, or providing feedback regarding said code generation goal." Applicants agree. To overcome the failings of Mantooth, the Office Action relies on the teachings Zink. In particular, the Office Action erroneously aligns the recited "a graphical user interface though which a user can specify at least one code generation goal" with the "wizards" of Zink that prompts the user for performance related information and then process the user inputs to determine actual property settings." Zink, col. 14, lines 2-5.

The combination of Mantooth in view of Zink and Iborra, as postulated by the Office Action, fails to establish a prima facie case of obviousness for at least the following reason.

Zink fails to teach or suggest "displaying a graphical user interface though which a user can **specify at least one code generation goal** for the embedded code to be generated from the model," as recited in claim 16. As discussed above, Zink discloses a system and method for graphical development of software and software/hardware hybrid systems. Zink, col. 1, lines 43-44. These systems may be represented by drawings which are comprised of components. Zink, col. 13, lines 48-49 and 57-59. However, Zink teaches **configuring the property settings for each component** on the drawing after the user creates the drawing, but before the drawing runs for the first time. Zink, col. 13, lines 60-62. "User access to property settings may be presented to the user in several different ways including [] 'wizards' that prompt the user for performance-related information and then process the user inputs to determine actual property settings." Zink, col. 13. line 66, to col. 14, line 5. In addition, Zink teaches including scripting language in components. Zink, col. 23, lines53-55. "Scripting provides capabilities for achieving compatibility (i.e. interoperability) as well as optimization based on user-defined goals. Specific capabilities (enabled via scripting) that

may be useful include elimination of parameter range-checking, elimination of data format conversions, re-linking selected code modules, patching code fragments into object modules, etc." Zink, col. 25, lines 1-7. "The scripting language permits the expert (who creates a component) to embed a significant measure of his (or her) integration knowledge into the component. Based on the script's ability to interrogate the operating environment, the expert can embed integration knowledge to select (or build) the appropriate code modules and contribute them to the solutions project." Zink, col. 23, lines 60-63. **Zink, therefore, teaches configuring the property settings for each component via a wizard or scripting language**, as opposed to "displaying a graphical user interface though which a user can **specify at least one code generation goal** for the embedded code to be generated from the model," as recited in claim 16. Mantooth and Iborra fail to overcome the failings of Zink. Hence, the combination of Mantooth, Zink, and Iborra fails to teach "displaying a graphical user interface though which a user can specify at least one code generation goal for the embedded code to be generated from the model," as recited in claim 16.

Dependent claims 17-29 are allowable, at least, for being dependent from an allowable claim.

Independent claim 30 recites similar subject matter to that recited in claim 16, which is allowable over Mantooth in view of Zink as discussed above. In addition, Iborra fails to overcome the failings of Mantooth and Zink. Therefore, claim 30 is allowable for the same reasons discussed above in connection with claim 16.

Independent claim 32 recites similar subject matter to that recited in claim 1, which is allowable over Mantooth, Zink, and Iborra as discussed above. Therefore, claim 32 is allowable for the same reasons discussed above in connection with claim 1.

Conclusion

All of the stated grounds of objection and rejection have been properly traversed, accommodated, or rendered moot. Applicants therefore respectfully request that the Examiner reconsider all presently outstanding objections and rejections and that they be withdrawn. Applicants believe that a full and complete reply has been made to the outstanding Office Action

and, as such, the present application is in condition for allowance. If the Examiner believes, for any reason, that personal communication will expedite prosecution of this application, the Examiner is hereby invited to telephone the undersigned at the number provided.

Prompt and favorable consideration of this Amendment is respectfully requested.

Respectfully submitted,

Dated: May 22, 2007

By ______________________

Michael A. Sartori, Ph.D.
Registration No.: 41,289
VENABLE LLP
P.O. Box 34385
Washington, DC 20043-9998
(202) 344-4000
(202) 344-8300 (Fax)
Attorney/Agent For Applicant

MAS/KDP
DC2/846676